

**ADOBE® FLASH® MEDIA  
INTERACTIVE SERVER 3.5  
プラグイン開発者ガイド**

© 2009 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Media Interactive Server 3.5 プラグイン開発者ガイド

本マニュアルがエンドユーザー使用許諾契約を含むソフトウェアと共に提供される場合、本マニュアルおよびその中に記載されているソフトウェアは、エンドユーザー使用許諾契約にもとづいて提供されるものであり、当該エンドユーザー使用許諾契約の契約条件に従ってのみ使用または複製することが可能となるものです。当該エンドユーザー使用許諾契約により許可されている場合を除き、本マニュアルのいかなる部分といえども、Adobe Systems Incorporated（アドビ システムズ社）の書面による事前の許可なしに、電子的、機械的、録音、その他いかなる形式・手段であれ、複製、検索システムへの保存、または伝送を行うことはできません。本マニュアルの内容は、エンドユーザー使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

本マニュアルに記載される内容は、あくまでも参照用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従って、当該情報が、アドビ システムズ社による確約として解釈されてはなりません。アドビ システムズ社は、本マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。

新しいアートワークを創作するためにテンプレートとして取り込もうとする既存のアートワークまたは画像は、著作権法により保護されている可能性のあるものであることをご留意ください。保護されているアートワークまたは画像を新しいアートワークに許可なく取り込んだ場合、著作権者の権利を侵害することがあります。従って、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名は、実在の組織を示すものではありません。

Adobe, the Adobe logo, and Flash are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/us/>

Portions include software under the following terms:

**Sorenson Spark** Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

Speech compression and decompression technology licensed from Nellymoser, Inc. ([www.nellymoser.com](http://www.nellymoser.com))

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are “Commercial Items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

# 目次

## プラグインの開発

バージョンングとアップグレード .....	1
プラグインの操作 .....	2
一般的な開発タスク .....	4
Access プラグインの開発 .....	5
Authorization プラグインの開発 .....	8
File プラグインの開発 .....	21
<b>索引</b> .....	<b>24</b>

# プラグインの開発

Adobe® Flash® Media Interactive Server および Adobe Flash Media Development Server は、C++ で作成されたプラグインアーキテクチャを提供します。このアーキテクチャによって、サーバーの機能を拡張することができます。拡張されたアクセス、権限およびファイル管理ソリューションを備えた固有の Flash Media Interactive Server および Flash Media Development Server のデプロイメントを構築するには、Access、Authorization および File プラグインを使用します。

例えば、アプリケーションレベルのコードに到達する前にクライアントの受諾、拒否またはリダイレクトを行うには、Access プラグインを使用することができます。また、ストリームやサーバーイベントへのアクセスを制御するには Authorization プラグインを、ファイル I/O メカニズムを作成するには File プラグインを使用することができます。

プラグインの API について詳しくは、『Adobe Flash Media Interactive Server Plug-in API リファレンスガイド』を参照してください。

## バージョンングとアップグレード

### バージョンングについて

Flash Media Server 3.5 以前にリリースされたプラグインのバージョン番号は 0.0 です。Flash Media Server 1.0 と同時にリリースされたプラグインのバージョン番号は 3.5 です。Flash Media Server 3.5 の時点では、File および Authorization プラグインに含まれている、バージョン番号を返す create および destroy メソッドが、次に示すように変更されています。

```
FmsCreateFileAdaptor2()  
FmsDestroyFileAdaptor2()  
FmsCreateAuthAdaptor2()  
FmsDestroyAuthAdaptor2()
```

サーバーは、これらのメソッドによって返された値に基づいて、File プラグインと Authorization プラグインのバージョン番号を検出します（Access プラグインには、Flash Media Server 3.5 に対応するバージョン番号はありません）。バージョン 3.0 のサーバーのプラグインを使用する場合、プラグインから返されるバージョン番号は 0.0 であり、バージョン 3.5 で使用可能な機能を使用することはできません。

バージョンング API について詳しくは、『Plug-in API リファレンスガイド』を参照してください。

**注意：**プラグインを Flash Media Server 3.0 機能に制限するには、FmsCreateXXXAdaptor2() への呼び出しからバージョン 0.0 を返します。

### プラグインのアップグレード

新しい API を含め、Flash Media Server 3.5 の新機能を使用するには、プラグインをアップグレードします。このリリースのサーバーの新機能を使用しない場合は、既存のプラグインコードを引き続き使用することができます。

- 1 Microsoft® Visual C++ 2003 または Microsoft Visual C++ 2005 を使用してプラグインを再コンパイルします。Linux では、GNU Compiler Collection 3.4.x を使用してプラグインを再コンパイルします。
- 2 File プラグインと Authorization プラグインでは、新しい FmsCreateXXXAdaptor2() メソッドと FmsDestroyXXXAdaptor2() メソッドを使用し、バージョン 1.0 を返します。『Plug-in API リファレンスガイド』を参照してください。

## プラグインの操作

### Flash Media Interactive Server 3.5 の新機能

Flash Media Interactive Server 3.5 には、プラグインに対する次の更新が含まれています。

- コアプロセスにアプリケーションを動的に割り当てるには、Access プラグインを使用します。7 ページの「[コアプロセスへのアプリケーションの割り当て](#)」を参照してください。
- SWF ファイルを検証するには、File プラグインを使用します。22 ページの「[検証のための外部 SWF ファイルの取得](#)」を参照してください。
- プラグインのバージョン情報を取得するには、Authorization プラグインと File プラグインを使用します。1 ページの「[バージョンングについて](#)」を参照してください。
- クライアント統計にアクセスするには、Authorization プラグインを使用します。11 ページの「[クライアント統計へのアクセス](#)」を参照してください。
- 動的ストリーム切り替えトランジションを処理するには、Authorization プラグインを使用します。12 ページの「[イベントとプロパティへのアクセス](#)」で、「F\_OLD\_STREAM\_NAME」、「F\_OLD\_STREAM\_TYPE」、「F\_OLD\_STREAM\_QUERY」、「F\_STREAM\_TRANSITION」の各プロパティを参照してください。
- DVR イベントを処理するには、Authorization プラグインを使用します。12 ページの「[イベントとプロパティへのアクセス](#)」で、「E\_RECORD」および「E\_RECORD\_STOP」イベントと、「F\_STREAM\_RECORD\_MAXSIZE」および「F\_STREAM\_RECORD\_DURATION」プロパティを参照してください。

### プラグインの開発とデプロイメントのワークフロー

- 1 組織のニーズに応じてサンプルのプラグインを変更したり、独自のプラグインを作成したりします。

2 ページの「[サンプルファイル](#)」を参照してください。

- 2 プラットフォーム用のプラグインの構築およびコンパイルを行います。

Windows でプラグインをコンパイルするには、Microsoft Visual Studio .NET 2003 または Microsoft Visual C++ 2005 を使用します。Linux でプラグインをコンパイルするには、GNU Compiler Collection 3.4.x を使用します。

プラグインを Release として構築します。プラグインを Debug として構築する場合、サーバープロセスではモジュールがロードされません。

プラグインは、Windows システムでは共有ライブラリ (DLL) ファイルとして、Linux システムでは共有オブジェクト (SO) ファイルとして実装されます。ファイル名は、libconnect.dll/libconnect.so、AuthModule.dll/AuthModule.so および FileModule.dll/FileModule.so となります。

64 ビットバージョンの linux GCC コンパイラに構築するには、GCC 用の -m32 スイッチと -B オプションを使用します。-B スイッチが /usr/lib32 ライブラリを指すようにします。

Access プラグインは、libconnect.dll/libconnect.so という名前にする必要があります。Authorization プラグインと File プラグインには任意の名前を付けることができます。

- 3 プラグインをデプロイします。

4 ページの「[プラグインのデプロイメント](#)」を参照してください。

### サンプルファイル

インストーラにより、各プラグイン用のサンプルファイルがインストールされます。これらのサンプルを使用して、API の使用方法を習得したり、独自のプラグインを作成するための素材にしたりすることもできます。

**注意:** デフォルトのプラグインを構築するには、ファイルの作成を使用します。ファイルの作成を実行するには、-f オプションと共にコマンドの作成を使用します。

### Access プラグイン

次のサンプルファイルが **RootInstall\samples\plugins\access** にインストールされます。

ファイル名	説明
adaptor.cpp sample.cpp adaptor.h	Access プラグインの C++ ファイルとヘッダーファイルのサンプル
AccessModuleSample.sln AccessModuleSample.vcproj	Windows 環境で Access プラグインを構築するために、Microsoft Visual C++ で使用する作業ファイル。ファイルの値を変更して、使用環境でのブラクティスを取り入れることができます。
Makefile.access	このファイルは Linux 上でデフォルトの Access プラグインを構築するために使用します。
StdAfx.h StdAfx.cpp	ファイルのインクルード宣言のサンプル

### Authorization プラグイン

次のサンプルファイルが **RootInstall\samples\plugins\auth** にインストールされます。

ファイル名	説明
AuthModule.cpp	Authorization プラグインの C++ ファイルのサンプル
AuthModule.sln AuthModule.vcproj	Windows 環境で Authorizaiton プラグインを構築するために、Microsoft Visual C++ で使用する作業ファイル。ファイルの値を変更して、使用環境でのブラクティスを取り入れることができます。
Makefile.AuthModule	このファイルは Linux 上でデフォルトの Authorization プラグインを構築するために使用します。
StdAfx.h StdAfx.cpp	ファイルのインクルード宣言のサンプル

### File プラグイン

次のサンプルファイルが **RootInstall\samples\plugins\file** にインストールされます。

ファイル名	説明
SimpleFileAdaptor.cpp SimpleFileAdaptor.h	File プラグインの C++ ファイルとヘッダーファイルのサンプル
FileModule.h	プラグインの create 関数と destroy 関数を定義するヘッダーファイル
FileUtil.cpp	ファイルとディレクトリを操作するためのユーティリティ関数
FileModule.sln FileModule.vcproj	Windows 環境で File プラグインを構築するために、Microsoft Visual C++ で使用する作業ファイル。ファイルの値を変更して、使用環境でのブラクティスを取り入れることができます。
Makefile.FileModule	このファイルは Linux 上でデフォルトの File プラグインを構築するために使用します。
StdAfx.h StdAfx.cpp	ファイルのインクルード宣言のサンプル

## ヘッダーファイル

ヘッダーファイル `FmsAdaptor.h`、`FmsAuthActions.h`、`FmsAuthAdaptor.h`、`FmsAuthEvents.h`、`FmsFileAdaptor.h`、`FmsMedia.h`、`IFCAccessAdaptor.h` はプラグイン API を定義しており、`RootInstall\samples\plugins\include` にインストールされています。

## プラグインのデプロイメント

デプロイされたプラグインは、サーバープロセスに組み込まれます。プラグインは、プロセスのスタートアップ中にロードされ、プロセスのシャットダウン中にアンロードされます。例えば、`vhost` が再起動と、新規コアプロセスが起動して `Authorization` プラグインのコピーがロードされます。`vhost` が停止すると、コアプロセスにより `Authorization` プラグインがアンロードされます。

- 1 サーバーを停止するには、次のいずれかの操作を行います。
  - Windows では、スタート/コントロールパネル/管理ツール/サービスを選択します。サービスの一覧から「Flash Media Server (FMS)」を選択し、「停止」をクリックします。
  - Linux では、シェルウィンドウを開いて、サーバーがインストールされているディレクトリ (`cd /opt/adobe/fms`) に移動します。`./server stop` と入力します。
- 2 コンパイル済みのプラグインの DLL ファイルまたは SO ファイルを次のいずれかのフォルダにコピーします。
  - `RootInstall/modules/access`
  - `RootInstall/modules/auth`
  - `RootInstall/modules/fileio`

**注意：**フォルダ名は変更しないでください。複数の `Authorization` プラグインをデプロイする場合は、すべてのプラグインを `/auth` フォルダにコピーしてください。

- 3 サーバーを起動するには、次のいずれかの操作を行います。
  - Windows では、スタート/コントロールパネル/管理ツール/サービスを選択します。サービスの一覧から「Flash Media Server (FMS)」を選択し、「開始」をクリックします。
  - Linux では、シェルウィンドウを開いて、サーバーがインストールされているディレクトリ (`cd /opt/adobe/fms`) に移動します。`./server start` と入力します。

## 一般的な開発タスク

### プラグインからログファイルへのデータの送信

`Authorization` および `File` プラグインからログファイルにカスタムメッセージを送信するには、プラグインで `IFmsServerContext` クラスの `log()` 関数を呼び出します。プラグインログファイルは `RootInstall/logs` ディレクトリにあり、デフォルトで `fileio.NN.log` および `authMessage.NN.log` という名前が付けられています。

**注意：** `Authorization` プラグインは、クライアント接続だけでなく、サーバーサイドの `Stream.play()` メソッドが呼び出されたときに行われる接続のログも記録します。これらの接続は、`access.log` ファイルと `authEvent.log` ファイルでは IP アドレス 127.0.0.1 によって区別されます。

`log()` 関数には、ログメッセージをシステムログ (Windows イベントビューアまたは Linux の `syslog`) にも記録するかどうかを指定する引数があります。デフォルト値は `false` です。システムログに過度にログを記録すると、パフォーマンスの問題を引き起こす可能性があります。

各ログファイルを有効または無効にするには、RootInstall/confにある Server.xml 設定ファイルの **Logging** セクションを編集します。ログファイルの場所と名前などの設定情報を追加または変更するには、**RootInstall/conf**にある Logger.xml ファイルを編集します。詳細については、XML ファイル内のコメントを参照してください。

Access プラグインは標準出力および標準エラーを閉じるプロセスによってホストされているので、プラグインの開発では、これらを利用してメッセージをログに書き込むことはできません。代わりに、Access プラグインの場合は、明示的にメッセージをファイルに書き込みます。サンプルの Access プラグインでは、回避策が示されています。

RootInstall/samples/plugins/access/adaptor.cpp ファイルで FILE\* logFile = fopen("SampleAdaptor.log", "a"); という行を検索してください。

## 設定ファイルからのデータの取得

Server.xml 設定ファイルに格納されたカスタムデータを取得するには、Authorization プラグインまたは File プラグインの IFmsServerContext クラスの getConfig() 関数を呼び出します。プラグインのこのデータを分析すると、サーバーの設定に応じてプラグインの異なる機能を有効にすることができます。

プラグインを使用すると、Plugins XML エlementに格納されたデータを取得することができます。カスタムデータを追加するには、RootInstall/conf フォルダにある Server.xml ファイルをテキストエディタで開き、データを追加します。XML を検証し、サーバーを再起動します。

getConfig() への呼び出しでは、テキストノード値のみを取得することができます。メソッドへの呼び出しでは、Element、複数のElement、またはネストされたElementとして追加された値を取得することができません。

IFmsServerContext->getConfig() の呼び出しの例については、AuthModule.cpp ファイルを参照してください。

## タイムクリティカルな呼び出しの処理

Flash Media Interactive Server または Flash Media Development Server から File プラグインまたは Authorization プラグインへの呼び出し（例えば、Authorization プラグインの authorize() など）は、すべてタイムクリティカルです。呼び出しが返されるまでサーバーは保留状態になるので、できるだけ速く呼び出しを処理し、サーバーに返す必要があります。呼び出しの処理に時間がかかり、待機またはスリープ操作が必要になった場合は、渡された引数を保持して、プラグインにより作成されたスレッドプールに渡す必要があります。例として、FileModule のサンプルを参照してください。

## Access プラグインの開発

### Access プラグインのアップグレード

以前のバージョンの Flash Media Server で Access プラグイン (libconnect.dll) を使用していて、最新の機能を使用したい場合は、そのプラグインを Microsoft Visual C++ 2003 または Microsoft Visual C++ 2005 を使用して再構築する必要があります。Linux では、GNU Compiler Collection 3.4.x を使用してプラグインを再構築する必要があります。また、インターフェイスが若干変更されています。

次の新しいクライアントプロパティは、Flash Media Interactive Server 3.5 以降および Flash Media Development Server 3.5 以降に搭載されています。

```
static const char* FLD_CORE_ID = "coreIdNum";  
static const char* FLD_PAGE_URL = "pageurl";
```

## Access プラグインの概要

Access プラグインによって、セキュリティの層がもう 1 つサーバーに追加されます。この層では接続要求をインターセプトし、要求がサーバーのスクリプト層に到達する前にクライアントとサーバーを検査して、要求の受諾、拒否またはリダイレクトのいずれの処理を行うかを決定します。使用できる Access プラグインは 1 つだけです。

サーバーに接続されているユーザー数や帯域幅の使用量などの基準に従って、接続要求の受諾、拒否またはリダイレクトを行うように、プラグインをコーディングすることができます。

また、組織のユーザーとパスワードのデータベースに対しクエリーを実行して、どの接続要求を許可するかを決定することができます。プラグインが接続を受諾すると、サーバーに対するそのユーザーのアクセス記録でデータベースを更新することができます。

サーバー上のファイルやフォルダに対する読み取りおよび書き込みアクセスの設定、オーディオやビデオビットマップデータに対するアクセス権限の設定およびクライアントプロパティの検査を行うことができます。

## Access プラグインの接続フロー

インストールされた Access プラグインは、サーバーの起動時にコンテキストポインタを使用して初期化されます。コンテキストポインタとプラグインポインタは、Access プラグインとサーバーの間に双方向通信を提供します。

クライアントがサーバーへの接続を試みると、サーバーは Access プラグインが存在するかどうかを確認します。Access プラグインが使用できる場合、このプラグインが接続要求を検査して、接続の許可、拒否またはリダイレクトを行います。Access プラグインが使用できない場合、接続要求は通常通り進行します。

**注意：**インストールされた Flash Media Interactive Server または Flash Media Development Server ごとに、Access プラグインを 1 つだけ使用できます。

## サーバーへの接続の書き換え

Access プラグインと Authorization プラグインはいずれも、サーバーへの接続を許可するために使用できます。ただし、それぞれのプラグインは異なる場所で実行され、接続プロセス内の異なる段階でアクティブになります。

新しい接続がエッジプロセス (fmsedge) に到達し、接続メッセージ内のデータは (エッジプロセスで動作する) Access プラグインに送信されます。この段階でプラグインは、IP アドレス、SWF ファイルの元の URL、接続 (または「ターゲット」) URI、ユーザーエージェントなど、クライアントに関する情報を保持しています。この情報を使用して、Access プラグインは接続の受諾、拒否またはリダイレクトを行ったり、ターゲット URI を書き換えたりすることができます。ターゲット URI を書き換えると、接続先が別の vhost、アプリケーションまたはアプリケーションインスタンスに強制的に変更されます。アダプタへのソケットレベルの接続が既に行われているので、プラグインは接続先を別のアダプタに書き換えることはできません。

外部からの接続の接続先を別の vhost、アプリケーションまたはアプリケーションインスタンスに強制的に変更するには、Access プラグインを使用する必要があります。Authorization プラグインに通知された時点では、ターゲット URI を書き換えるには遅すぎます。また、Access プラグインは接続プロセスの早い段階で動作するので、接続をスクリーニングするには、このプラグインを使用するのが最も効率的です。許可の動作をできるだけ軽量にするためには、接続 URI を書き換える必要がなくても、Access プラグインを使用してください。

Authorization プラグインは、接続がアプリケーションに転送された後は、コアプロセス (fmscore) で動作します。Authorization プラグインは、クライアントによるアプリケーションへの接続試行を受諾、拒否またはリダイレクトすることができますが、接続先を別の URI に書き換えることはできません。

**注意：**接続のリダイレクトは、接続 URI の書き換えとは異なります。クライアントをリダイレクトすると、新しい URI を含むリダイレクトメッセージがクライアントに返されて、現在の接続が終了します。すると、クライアントは新しい URI への接続を試みます。

## Access プラグインのコードの記述

アドビが提供したサンプルの Access プラグインファイルのコードを変更したり、独自のプラグインを作成したりすることができます。

クライアント接続要求は、Access プラグインの `onAccess()` コールバック関数をトリガします。接続要求を検査し、クライアントプロパティを変更し、`onAccess()` コールバック関数による要求を受諾、拒否、またはリダイレクトするコードを記述します。

クライアントフィールドに対するクエリーを実行するには、`getValue()` を呼び出します。クライアントフィールドを変更するには、`setValue()` を呼び出します。フィールドの一覧については、[www.adobe.com/go/documentation](http://www.adobe.com/go/documentation) にある『**Adobe Flash Media Interactive Server Plug-in API** リファレンスガイド』の「`fms_access` 名前空間」を参照してください。接続元の SWF ファイルまたはサーバーの URL であるフィールド `x-page-url` が、Flash Media Server 3.5 に追加されました。

サーバーの統計 `eTOTAL_CONNECTED`、`eBYTES_IN` および `eBYTES_OUT` に対してクエリーを実行するには、`getStats()` を呼び出します。

フィールドに対してクエリーを実行し、接続ロジックを記述した場合は、`accept()`、`reject()` または `redirect()` を呼び出して、サーバー上のアプリケーションに対するクライアントの接続を許可または拒否します。

## コアプロセスへのアプリケーションの割り当て

コアプロセスにはアプリケーションを動的に割り当てることができます。この機能は、CPU 消費量および記録済みメディアのキャッシュサイズなどのリアルタイムのパフォーマンスカウンタに基づいて、複数のコアプロセスに負荷を分散する場合に使用します。この機能は、特定のユーザーにに対して、高いサービス品質 (QoS) を提供する場合にも使用することができます。

どのコアプロセスがアプリケーションを処理するかを指定するには、「`coreIdNum`」フィールドとコア番号を `setValue()` 関数に渡します。コア番号は任意の正の整数でありえます。サーバーは `core_number % number_of_cores` という式を使用して、使用するコアプロセスを決定します。例えば、次のコードでは、コア番号 3 が割り当てられます。

```
char* coreId = "3";
if(!setValue("coreIdNum", coreId)) {
    FILE * pFile = fopen ("error.log","a");
    fprintf(pFile, "Core id = %s", coreId);
}
```

次のように `getValue()` を呼び出すと、クライアント接続が割り当てられているコアプロセスを特定することができます。

```
const char* coreId = getValue("coreIdNum");
FILE * pFile = fopen ("output.txt","a");
fprintf(pFile, "Core id = %s", coreId);
```

### 負荷分散

Access プラグインを使用して、CPU の使用状況や記録メディアのキャッシュサイズなど、パフォーマンス基準を監視し、それらの基準に応じて負荷分散を実行するようにコーディングできます。複数のコアプロセスに対するアプリケーションの分散は、`Application.xml` の `numprocs` および `scope` エlement で定義されます。例えば、`numprocs` を 3 に、`scope` を `application` に設定すると、各アプリケーションでは 3 つのコアプロセスで外部からの接続が処理されます。コアプロセスが過負荷である場合は、過負荷になっているコアプロセスに外部からの新たな接続が送信されないようにする、Access プラグインを開発することができます。1 つの方法として、次のことを実行するプラグインコードを記述します。

- 1 各コアプロセスの統計を追跡します。Administration API である `getFileCacheStats()`、`getAppStats()`、`getInstanceStats()` を使用します。
- 2 新しいクライアントが接続した場合には、コアプロセスが過負荷になっていないかどうかをプラグインコードで判断する必要があります。過負荷になっている場合は、過負荷になっていない別のプロセスのコア ID をプラグインコードで判断し、新しい接続をそのプロセスに送信します。

## サービス品質 (QoS)

特定のクライアントに対する QoS を高めるには、そのクライアントだけに特定のコアプロセスを使用することができます。例えば、コンテンツへのアクセスについてプレミアムサブスクリプション料金を支払っている特定のクライアントに、可能な限り最高のパフォーマンスを提供するとします。この場合は、6つのコアプロセスにアプリケーションを分散して各プロセスの統計を監視する Access プラグインを開発できます。クライアントによっては、異なるコアプロセスがアプリケーションに対する接続を処理します。3つのコアプロセスが上位のクライアントに対して予約され、残り3つのコアプロセスがその他すべてのクライアントに割り当てられます。このプラグインは、各プロセスの統計を監視します。特定のプロセスが過負荷になると、Access プラグインによって、過負荷になっているプロセスが新しい接続を処理することが防止されます。このプラグインは、処理能力があるプロセスに新しい接続をルーティングできます。

## サーバーのフォルダ権限の設定

Application.xml 設定ファイルの Access セクションを使用すると、Access プラグインのフォルダレベルの権限を設定できます。この設定ファイルは、各仮想ホストディレクトリにあります。また、アプリケーションディレクトリにもある可能性があります。FolderAccess エlementが true に設定されている場合、Access プラグインの readAccess および writeAccess フィールドを使用して個々のファイルに対する権限を設定することはできません。設定できるのはフォルダレベルの権限だけです。デフォルト値は false で、次のように個々のファイルに対する権限を設定することができます。

```
<!-- Controls libconnect.dll access configurations -->
<Application>
  ...
  <Client>
    ...
    <Access>
      <FolderAccess>false</FolderAccess>
    </Access>
    ...
  ...
</Application>
```

# Authorization プラグインの開発

## Authorization プラグインの概要

Authorization プラグインはクライアントに対し、サーバーイベントと、それらのイベントに関連付けられているフィールドへのアクセスを許可します。Authorization プラグインを使用すると、次のことを実行できます。

- サーバーへの接続の許可

**注意：** Access プラグインも同様に接続を許可するために使用でき、動作はより軽くなります。詳細については、6ページの「[Access プラグインの概要](#)」を参照してください。

- ストリームの再生またはストリーム内のシークの許可
- 動的ストリームトランジションの許可
- ストリームのパブリッシュまたはストリームの記録の許可
- 物理的な場所への論理 URL のマッピング

例えば、ビデオ Player でストリーム「foo」を再生する場合 (ns.play("myvideos/foo")), サーバーが要求を処理するときこの仮想名を c:¥apps¥vidapp¥myvideos¥ にマッピングすることができます。Authorization プラグインを使用すると、これを別の物理的な場所 (c:¥myvideos¥ など) に再マッピングすることができます。

- サーバーからのクライアントの切断

- サーバーサイド ActionScript 内のメソッドの呼び出し
- クライアント統計へのアクセス
- パブリッシュしているストリーム内で新しいコーデックが検出されたときの識別
- サイズまたは継続時間による記録サイズの制限

次の 1 つ以上の基準に従ってコンテンツをクライアントに配信するには、**Authorization** プラグインを使用します。

- 地理的な場所
- サブスクリプションレベル
- ストリームのオリジン
- 特定のストリームに対するユーザーのアクセスの時刻と継続時間

**Authorization** プラグインを使用して、ストリームのサービス品質 (QoS) を監視することもできます。プラグインはライブストリームの QoS 情報を外部のログファイルに報告し、これをさらに別のカスタム構築されたツールに読み込むこともできます。

## 複数の Authorization プラグインの使用

外部からのイベントに対して順次アクションを実行するために、一連のプラグインを使用することができます。個々のプラグインに特定のタイプのイベントを割り当てます。例えば、`auth1.dll` (または `auth1.so`) でストリームの再生を許可し、`auth2.dll` (または `auth2.so`) でストリームのパブリッシュを許可するなどの割り当てを行います。

**注意:** プラットフォーム間の互換性を維持するため、名前には小文字を使用します。

サーバーはプラグインをアルファベット順にロードします。サーバーは、クライアントによるイベント要求を、アルファベット順に従って処理します。各プラグインは外部からの要求をフィルタ処理します。

## Authorization プラグインのコードの記述

サーバーは **Authorization** プラグインをロードする際に、次のエントリポイントを予期しています。

```
FmsCreateAuthAdaptor() // Creates an Authorization plug-in.  
FmsDestroyAuthAdaptor() // Destroys an Authorization plug-in.
```

または

```
FmsCreateAuthAdaptor2() // Creates an Authorization plug-in with versioning information.  
FmsDestroyAuthAdaptor2() // Destroys an Authorization plug-in with versioning.
```

`FmsCreateAuthAdaptor2()` を呼び出して、プラグインのバージョン番号を検出します。バージョン番号を検出すると、プラグインがバージョン 3.5 の機能をサポートすることを確認することができます。プラグインの旧バージョンでは、`E_CODEC_CHANGE`、`E_RECORD`、`E_RECORD_STOP` イベントは使用できません。

イベントを発生時に処理するには、`IFmsAuthAdaptor` インターフェイスを実装します。このクラスには、`authorize()`、`notify()` および `getEvents()` 関数が含まれます。

プラグインが作成されると、サーバーは `getEvents()` 関数を呼び出し、プラグインがどのイベントを処理するかを確認します。サーバーが `getEvents()` を一度呼び出すと、イベントはプラグインの存続期間にわたって有効となります。

**重要:** `E_CODEC_CHANGE` イベントが除外されていない場合、プラグインでは、すべてのメッセージをスキャンしてコーデックの変更を検出します。必要な場合にのみ、このイベントをサブスクライブします。このイベントは、サーバーと共にインストールされた **Authorization** プラグインのサンプルコードでは無効です。

一部のイベントは、サーバーで実行する前にプラグインによって許可される必要がありますが、その他のイベントは、プラグインがイベントの通知を受け取ったことをサーバーに通知するだけで済みます。許可イベントが発生すると、サーバーはプラグインの `authorize()` 関数を呼び出します。通知イベントが発生すると、サーバーはプラグインの `notify()` 関数を呼び出します。これらの関数で、イベントが発生したときに実行されるコードを記述します。

サーバーは、プラグインが `IFmsAuthServerContext` クラスの `onAuthorize()` 関数を呼び出すまで操作を中断します。`authorize()` 関数の最終行で `onAuthorize()` を呼び出します。その関数に、イベントに対するポイントと、イベントが許可されているかどうかを示すブール値を渡します。この呼び出しにより、サーバー上で保留中になっていた操作が完了します。

`IFmsAuthServerContext` クラスの `notify()` 関数を呼び出すと、サーバーの操作が続行されます。`notify()` 関数の最終行で `onNotify()` を呼び出します。イベントに対するポイントを渡します。この呼び出しにより、通知が完了してイベントが不要になったことが、サーバーに通知されます。

イベントを処理する方法には、アクションをイベントに割り当てる方法と、`authorize()` 関数と `notify()` 関数にイベントを処理するコードを記述する方法の2つがあります。

### イベントへのアクションの割り当て

イベントには、`IFmsDisconnectAction` および `IFmsNotifyAction` の2つのアクションを割り当てることができます。

`IFmsDisconnectAction` はイベントの発生時に1つ以上のクライアントを切断し、`IFmsNotifyAction` はイベントの発生時に、サーバーサイドスクリプト内の `Client` オブジェクトまたはアプリケーションオブジェクトに対する関数を呼び出します。追加できるアクションの数に制限はありません。アクションは、イベントが処理される直前に割り当てられる順序で実行されます。

イベントにアクションを割り当てるには、`addDisconnectAction()` または `addNotifyAction()` を `IFmsAuthEvent` インスタンスから呼び出します。アクションは、イベントが処理される直前に実行されます。イベントに複数のアクションを割り当てると、アクションは割り当てられた順序で実行されます。

次のコードでは、`IFmsNotifyAction` インスタンス `pAction` を `IFmsAuthEvent` インスタンス `m_pAev` に追加しています。このアクションは、サーバーサイドスクリプト内の `someMethod()` を呼び出し、パラメータ `12345` を渡します。

```
FmsVariant field;
IFmsNotifyAction* pAction = m_pAev->addNotifyAction("Notified by adaptor");
pAction->setClientId(field);
field.setString("someMethod");
pAction->setMethodName(field);
field.setU16(12345);
pAction->addParam(field);
```

`IFmsNotifyAction` インスタンスの `addParam()` を呼び出して、パラメータをメソッドに渡すことができます。サーバーサイドスクリプトでメソッドを定義する必要があります。`setClientId()` に値を渡すと、サーバーサイド `Client` オブジェクトでメソッドが呼び出されます。`setClientId()` に値を渡さなかった場合、サーバーサイドアプリケーションオブジェクトでメソッドが呼び出されます。例えば、次のサーバーサイド `ActionScript` コードは、`Client` オブジェクトに対する `someMethod()` を定義しています。

```
application.onConnect = function(client){
    client.someMethod = function(msg){
        trace("inside someMethod");
    }
}
```

### ストリームパスのマッピング

`authorize()` 関数では、仮想ストリームパスを別の物理的な場所に再マッピングする必要がある場合があります。例えば、クライアント1とクライアント2の両方がストリーム `foo.flv` の再生を要求した場合、2つの `E_FILENAME_TRANSFORM` イベントが呼び出されますが、クライアントごとに個別にストリームを再マッピングすることができます。例えば、クライアント1ではストリーム `foo` を `c:¥yourpath1¥foo.flv` に再マッピングし、クライアント2ではストリーム `foo` を `c:¥yourpath2¥foo.flv` に再マッピングすることができます。

論理ストリームを別の物理パスにマッピングするには、Application.xml ファイルで次のようにパラメータを設定する必要があります。

```
<Application>
  ...
  <StreamManager>
    ...
    <QualifiedStreamsMapping enable="true" />
  
```

クライアントがストリームを再生またはパブリッシュしたり、サーバーサイドの ActionScript でストリーム呼び出しを実行したりする場合には必ず、E\_FILENAME\_TRANSFORM イベントが発生します。E\_FILENAME\_TRANSFORM イベントを使用すると、F\_STREAM\_PATH プロパティを変更して、ストリームの物理的な場所のパスを変更することができます。

E\_FILENAME\_TRANSFORM イベントは、Authorization プラグインと File プラグインの間のリンクです。このイベントで指定するストリームの物理的な場所へのパスは、特定のファイルの操作で使用するために File プラグインに渡されます。

イベントは次の順に発生します。

- 1 E\_PLAY または E\_PUBLISH 論理ストリームの名前をここで変更できます。
- 2 E\_FILENAME\_TRANSFORM 結果として生じる物理ストリームの名前をここで変更することがあります。
- 3 再生が行われます。
- 4 E\_STOP または E\_UNPUBLISH

#### エッジサーバーおよびオリジンサーバーのデプロイメントの違い

Authorization プラグインはオリジンサーバーとエッジサーバーにデプロイすることができますが、それぞれの場合で機能が異なります。

**オリジン** Authorization プラグインがオリジンサーバーにインストールされている場合、E\_PLAY (ライブストリーム用) イベントまたは E\_LOADSEGMENT (記録ストリーム用) イベントがクリップを再生するために呼び出されます。記録ストリームの再生を阻止するには、E\_PLAY イベントではなく E\_LOADSEGMENT イベントを処理します。

**エッジ (コア)** サーバーサイド ActionScript はエッジサーバーでは実行されないため、NOTIFY アクションは使用できず、常に失敗のステータスが返されます。

#### エラーの処理

Authorization プラグインは、FMSCore プロセスによってロードされます。プラグイン内で例外が発生すると、FMSCore がクラッシュします。

**重要：** Authorization プラグインは、例外を処理して FMSCore のクラッシュを防止する必要があります。

## クライアント統計へのアクセス

C++ 層の Authorization プラグインを通じてクライアント統計にアクセスすると、サーバーサイドスクリプト内のクライアント統計にアクセスすることより、サーバーのパフォーマンスが向上します。Authorization プラグインは、要求を集約してからサーバーに送信します。

次のクライアント統計は、FmsAuthAdaptor.h ファイルの FmsClientStats 構造体で定義されます。

統計	説明
bytes_in	受信した合計バイト数。
bytes_out	送信した合計バイト数。

統計	説明
msg_in	受信したメッセージの合計数。
msg_out	送信したメッセージの合計数。
msg_dropped	欠落したメッセージの合計数。

クライアント統計にアクセスするには、FmsCreateAuthAdaptor2() 関数を実装します。その例については、**RootInstall\samples\plugins\auth** にインストールされている AuthModule.cpp ファイルを参照してください。

クライアントがサーバーに接続すると、サーバーは E\_CONNECT イベントを Authorization プラグインに送信します。Authorization プラグインが E\_CONNECT イベントを受信した後で、IFmsAuthServerContext::getClientStats() を呼び出して統計を取得することができます。E\_PLAY や E\_STOP など、すべてのイベントには、F\_CLIENT\_STATS\_HANDLE というフィールドがあります。このクライアントの構造ハンドルを getClientStats() 呼び出しでサーバーに戻します。E\_CONNECT および E\_DISCONNECT 以外の通知イベント中に F\_CLIENT\_STATS\_HANDLE を使用する場合は、ハンドルを保存する必要はありません。

クライアントが接続していることを確認するには、getClientStats() で呼び出しの戻りステータスを確認します。この確認は、特に複数の Authorization プラグインを使用している場合に必要になります。例えば、プラグイン A が E\_CONNECT イベントにサブスクライブし、プラグイン B が E\_DISCONNECT イベントにサブスクライブするとします。クライアントとサーバーとの接続が切断されると、アダプタ A は E\_DISCONNECT メッセージを受信しません。

サーバーはクライアントから切断メッセージを受信すると、E\_DISCONNECT イベントで Authorization プラグインに通知します。F\_CLIENT\_STATS\_HANDLE フィールドは、E\_DISCONNECT イベントの後に無効になります。

## イベントとプロパティへのアクセス

サーバーはインターフェイスをイベントオブジェクト IFmsAuthEvent に提供します。これにより、プラグインはイベントにアクセスすることができます。プラグインは各イベントの間に、特定のクライアント、アプリケーションおよびストリームのプロパティにアクセスすることができます。すべてのイベントの間にすべてのプロパティにアクセスできるわけではありません。プロパティはそれぞれ読み取り専用または読み取りと書き込み可能になっています。次の表は、プロパティとイベントを一覧にしたものです。見出しは、イベントが通知、許可またはその双方を必要とするかどうかを示しています。

E\_CODEC\_CHANGE、E\_RECORD および E\_RECORD\_STOP イベントは、Flash Media Server 3.5 の新しいイベントです。

**注意：**表の幅が広すぎてページ内に収まらないので、この情報は複数の表に分けられています。

	サーバーのパラメータ	E_APPSTART 通知	E_APPSTOP 通知	E_CONNECT 通知および許可
F_CLIENT_URI	2.x	なし	なし	読み取り専用
F_CLIENT_REDIRECT_URI	3	なし	なし	読み取りと書き込み
F_CLIENT_ID	2.x	なし	なし	読み取り専用
F_CLIENT_IP	2.x	なし	なし	読み取り専用
F_CLIENT_SECURE	2.x	なし	なし	読み取り専用
F_CLIENT_VHOST	2.x	なし	なし	読み取り専用
F_CLIENT_REFERRER	2.x	なし	なし	読み取り専用
F_CLIENT_PAGE_URL	2.x	なし	なし	読み取り専用
F_CLIENT_AMF_ENCODING	2.x	なし	なし	読み取り専用

	サーバーのバージョン	E_APPSTART 通知	E_APPSTOP 通知	E_CONNECT 通知および許可
F_CLIENT_USER_AGENT	2.x	なし	なし	読み取り専用
F_CLIENT_READ_ACCESS	2.x	なし	なし	読み取りと書き込み
F_CLIENT_WRITE_ACCESS	2.x	なし	なし	読み取りと書き込み
F_CLIENT_READ_ACCESS_LOCK	2.x	なし	なし	読み取りと書き込み
F_CLIENT_WRITE_ACCESS_LOCK	2.x	なし	なし	読み取りと書き込み
F_CLIENT_AUDIO_SAMPLE_ACCESS	2.x	なし	なし	読み取りと書き込み
F_CLIENT_VIDEO_SAMPLE_ACCESS	2.x	なし	なし	読み取りと書き込み
F_CLIENT_AUDIO_SAMPLE_ACCESS_LOCK	2.x	なし	なし	読み取りと書き込み
F_CLIENT_VIDEO_SAMPLE_ACCESS_LOCK	2.x	なし	なし	読み取りと書き込み
F_CLIENT_AUDIO_CODECS	2.x	なし	なし	読み取り専用
F_CLIENT_VIDEO_CODECS	2.x	なし	なし	読み取り専用
F_CLIENT_TYPE	2.x	なし	なし	読み取り専用
F_CLIENT_PROTO	2.x	なし	なし	読み取り専用
F_CLIENT_URI_STEM	2.x	なし	なし	読み取り専用
F_CLIENT_STATS_HANDLE	3.5	なし	なし	読み取り専用
F_APP_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_INST	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_NAME	2.x	なし	なし	なし
F_STREAM_PATH	2.x	なし	なし	なし
F_STREAM_TYPE	2.x	なし	なし	なし
F_STREAM_LENGTH	2.x	なし	なし	なし
F_STREAM_POSITION	2.x	なし	なし	なし
F_STREAM_IGNORE	2.x	なし	なし	なし
F_STREAM_RESET	2.x	なし	なし	なし
F_STREAM_QUERY	2.x	なし	なし	なし
F_STREAM_PAUSE	3	なし	なし	なし
F_STREAM_PAUSE_TIME	3	なし	なし	なし
F_STREAM_PAUSE_TOGGLE	3	なし	なし	なし
F_STREAM_SEEK_POSITION	3	なし	なし	なし
F_STREAM_PUBLISH_TYPE	3	なし	なし	なし
F_STREAM_PUBLISH_BROADCAST	3	なし	なし	なし

	サーバーのバージョン	E_APPSTART 通知	E_APPSTOP 通知	E_CONNECT 通知および許可
F_SEGMENT_START	3	なし	なし	なし
F_SEGMENT_END	3	なし	なし	なし
F_STREAM_ID	3.5	なし	なし	なし
F_STREAM_CODEEC	3.5	なし	なし	なし
F_STREAM_CODEEC_TYPE	3.5	なし	なし	なし
F_OLD_STREAM_NAME	3.5	なし	なし	なし
F_OLD_STREAM_QUERY	3.5	なし	なし	なし
F_OLD_STREAM_TYPE	3.5	なし	なし	なし
F_STREAM_TRANSITION	3.5	なし	なし	なし
F_STREAM_RECORD_MAXSIZE	3.5	なし	なし	なし
F_STREAM_RECORD_MAXDURATION	3.5	なし	なし	なし

以下は追加イベントです。

	サーバーのバージョン	E_DISCONNECT 通知	E_FILENAME_TRANSFORM 通知および許可	E_PLAY 通知および許可
F_CLIENT_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REDIRECT_URI	3	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_ID	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_IP	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_SECURE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VHOST	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REFERER	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PAGE_URL	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AMF_ENCODING	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_USER_AGENT	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用

	サーバーのバージョン	E_DISCONNECT 通知	E_FILENAME_TRANSFORM 通知および許可	E_PLAY 通知および許可
F_CLIENT_VIDEO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PROTO	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PROTO	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_STATS_HANDLE	3.5	読み取り専用	読み取り専用	読み取り専用
F_APP_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_INST	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_NAME	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_PATH	2.x	なし	読み取りと書き込み	読み取り専用
F_STREAM_TYPE	2.x	なし	読み取りと書き込み	読み取りと書き込み
F_STREAM_LENGTH	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_POSITION	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_IGNORE	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_RESET	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_QUERY	2.x	なし	読み取り専用	読み取りと書き込み
F_STREAM_PAUSE	3	なし	なし	なし
F_STREAM_PAUSE_TIME	3	なし	なし	なし
F_STREAM_PAUSE_TOGGLE	3	なし	なし	なし
F_STREAM_SEEK_POSITION	3	なし	なし	なし
F_STREAM_PUBLISH_TYPE	3	なし	なし	なし
F_STREAM_PUBLISH_BROADCAST	3	なし	なし	なし
F_SEGMENT_START	3	なし	なし	なし
F_SEGMENT_END	3	なし	なし	なし
F_STREAM_ID	3.5	なし	読み取り専用	読み取り専用
F_STREAM_CODEC	3.5	なし	なし	なし
F_STREAM_CODEC_TYPE	3.5	なし	なし	なし
F_OLD_STREAM_NAME	3.5	なし	なし	読み取りと書き込み
F_OLD_STREAM_QUERY	3.5	なし	なし	読み取りと書き込み
F_OLD_STREAM_TYPE	3.5	なし	なし	読み取りと書き込み

	サーバーのバージョン	E_DISCONNECT 通知	E_FILENAME_TRANSFORM 通知および許可	E_PLAY 通知および許可
F_STREAM_TRANSITION	3.5	なし	なし	読み取りと書き込み
F_STREAM_RECORD_MAXSIZE	3.5	なし	なし	なし
F_STREAM_RECORD_MAXDURATION	3.5	なし	なし	なし

次の表には、追加イベントが含まれています。

	サーバーのバージョン	E_PUBLISH 通知および許可	E_STOP 通知	E_UNPUBLISH 通知
F_CLIENT_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REDIRECT_URI	3	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_ID	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_IP	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_SECURE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VHOST	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REFERER	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PAGE_URL	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AMF_ENCODING	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_USER_AGENT	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PROTO	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_URI_STEM	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_STATS_HANDLE	3.5	読み取り専用	読み取り専用	読み取り専用
F_APP_URI	2.x	読み取り専用	読み取り専用	読み取り専用

	サーバーのバージョン	E_PUBLISH 通知および許可	E_STOP 通知	E_UNPUBLISH 通知
F_APP_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_INST	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_NAME	2.x	読み取りと書き込み	読み取り専用	読み取り専用
F_STREAM_PATH	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_TYPE	2.x	読み取りと書き込み	読み取り専用	読み取り専用
F_STREAM_LENGTH	2.x	読み取りと書き込み	読み取り専用	読み取り専用
F_STREAM_POSITION	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_IGNORE	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_RESET	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_QUERY	2.x	読み取りと書き込み	読み取り専用	読み取り専用
F_STREAM_PAUSE	3	なし	なし	なし
F_STREAM_PAUSE_TIME	3	なし	なし	なし
F_STREAM_PAUSE_TOGGLE	3	なし	なし	なし
F_STREAM_SEEK_POSITION	3	なし	なし	なし
F_STREAM_PUBLISH_TYPE	3	読み取りと書き込み	なし	読み取り専用
F_STREAM_PUBLISH_BROADCAST	3	読み取りと書き込み	なし	読み取り専用
F_SEGMENT_START	3	なし	なし	なし
F_SEGMENT_END	3	なし	なし	なし
F_STREAM_ID	3.5	読み取り専用	読み取り専用	読み取り専用
F_STREAM_CODEEC	3.5	なし	なし	なし
F_STREAM_CODEEC_TYPE	3.5	なし	なし	なし
F_OLD_STREAM_NAME	3.5	なし	なし	なし
F_OLD_STREAM_QUERY	3.5	なし	なし	なし
F_OLD_STREAM_TYPE	3.5	なし	なし	なし
F_STREAM_TRANSITION	3.5	なし	なし	なし
F_STREAM_RECORD_MAXSIZE	3.5	なし	なし	なし
F_STREAM_RECORD_MAXDURATION	3.5	なし	なし	なし

次の表には、追加イベントが含まれています。

	サーバーのパラメータ	E_SEEK 通知および許可	E_PAUSE 通知および許可	E_LOADSEGMENT 通知および許可
F_CLIENT_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REDIRECT_URI	3	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_ID	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_IP	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_SECURE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VHOST	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REFERER	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PAGE_URL	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AMF_ENCODING	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_USER_AGENT	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PROTO	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_URI_STEM	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_STATS_HANDLE	3.5	読み取り専用	読み取り専用	読み取り専用
F_APP_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_INST	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_PATH	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_LENGTH	2.x	読み取りと書き込み	読み取り専用	読み取り専用
F_STREAM_POSITION	2.x	読み取り専用	読み取り専用	読み取り専用

	サーバーのバージョン	E_SEEK 通知および許可	E_PAUSE 通知および許可	E_LOADSEGMENT 通知および許可
F_STREAM_IGNORE	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_RESET	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_QUERY	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_PAUSE	3	なし	読み取り専用	なし
F_STREAM_PAUSE_TIME	3	なし	読み取り専用	なし
F_STREAM_PAUSE_TOGGLE	3	なし	読み取り専用	なし
F_STREAM_SEEK_POSITION	3	読み取りと書き込み	なし	なし
F_STREAM_PUBLISH_TYPE	3	なし	なし	なし
F_STREAM_PUBLISH_BROADCAST	3	なし	なし	なし
F_SEGMENT_START	3	なし	なし	読み取り専用
F_SEGMENT_END	3	なし	なし	読み取り専用
F_STREAM_ID	3.5	読み取り専用	読み取り専用	読み取り専用
F_STREAM_CODEC	3.5	なし	なし	なし
F_STREAM_CODEC_TYPE	3.5	なし	なし	なし
F_OLD_STREAM_NAME	3.5	なし	なし	なし
F_OLD_STREAM_QUERY	3.5	なし	なし	なし
F_OLD_STREAM_TYPE	3.5	なし	なし	なし
F_STREAM_TRANSITION	3.5	なし	なし	なし
F_STREAM_RECORD_MAXSIZE	3.5	なし	なし	なし
F_STREAM_RECORD_MAXDURATION	3.5	なし	なし	なし

次の表には、追加イベントが含まれています。

プロパティ	サーバーのバージョン	E_CODEC_CHANGE 通知	E_RECORD 通知および許可	E_RECORD_STOP 通知
F_CLIENT_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REDIRECT_URI	3	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_ID	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_IP	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_SECURE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VHOST	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_REFERER	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PAGE_URL	2.x	読み取り専用	読み取り専用	読み取り専用

プロパティ	サーバーのバージョン	E_CODEC_CHANGE 通知	E_RECORD 通知および許可	E_RECORD_STOP 通知
F_CLIENT_AMF_ENCODING	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_USER_AGENT	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_READ_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_WRITE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_SAMPLE_ACCESS_LOCK	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_AUDIO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_VIDEO_CODECS	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_PROTO	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_URI_STEM	2.x	読み取り専用	読み取り専用	読み取り専用
F_CLIENT_STATS_HANDLE	3.5	読み取り専用	読み取り専用	読み取り専用
F_APP_URI	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_APP_INST	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_NAME	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_PATH	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_TYPE	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_LENGTH	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_POSITION	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_IGNORE	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_RESET	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_QUERY	2.x	読み取り専用	読み取り専用	読み取り専用
F_STREAM_PAUSE	3	なし	読み取り専用	なし
F_STREAM_PAUSE_TIME	3	なし	読み取り専用	なし
F_STREAM_PAUSE_TOGGLE	3	なし	読み取り専用	なし
F_STREAM_SEEK_POSITION	3	なし	読み取り専用	なし
F_STREAM_PUBLISH_TYPE	3	読み取り専用	読み取り専用	読み取り専用

プロパティ	サーバーのバージョン	E_CODEC_CHANGE 通知	E_RECORD 通知および許可	E_RECORD_STOP 通知
F_STREAM_PUBLISH_BROADCAST	3	読み取り専用	読み取り専用	読み取り専用
F_SEGMENT_START	3	なし	なし	なし
F_SEGMENT_END	3	なし	なし	なし
F_STREAM_ID	3.5	読み取り専用	読み取り専用	読み取り専用
F_STREAM_CODEC	3.5	読み取り専用	なし	なし
F_STREAM_CODEC_TYPE	3.5	読み取り専用	なし	なし
F_OLD_STREAM_NAME	3.5	なし	なし	なし
F_OLD_STREAM_QUERY	3.5	なし	なし	なし
F_OLD_STREAM_TYPE	3.5	なし	なし	なし
F_STREAM_TRANSITION	3.5	なし	なし	なし
F_STREAM_RECORD_MAXSIZE	3.5	なし	読み取りと書き込み	読み取り専用
F_STREAM_RECORD_MAXDURATION	3.5	なし	読み取りと書き込み	読み取り専用

## File プラグインの開発

### File プラグインの概要

File プラグインを使用すると、サーバーがファイルシステムからコンテンツを読み取る場所と方法を制御することができます。このプラグインは、オペレーティングシステムのファイル I/O メカニズムとサーバーの間のインターフェイスを提供します。サンプルファイルの設定または変更を行うと、デフォルトのオペレーティングシステムに基づくファイルシステム I/O に代わるものを作成することができます。

サーバーの以前のバージョンでは、ファイルシステムへの同期アクセスをサポートしていました。ファイルの読み取り操作を要求したときには、そのたびにキューにあるそれより前の要求が完了するまで待つ必要がありました。File プラグインは非同期アクセスをサポートしており、ネットワークベースのファイル I/O をより簡単に実装できるようになっています。

次のことを実行するように File プラグインをコーディングすることができます。

- HTTP を介してリモートからファイルを取得し、コアサーバープロセスを通じてそれらのファイルをクライアントに配信することで、コンテンツの管理作業の負荷を軽減します。
- 別の物理的な場所にファイルを再マッピングします。
- 外部 SWF ファイルを取得して検証します。

**注意：** File プラグインはストリームファイルと SWF ファイルで使用します。

カスタム File プラグインが存在しないか、またはアクティブになっていない場合、サーバーは下位互換性のために標準のオペレーティングシステムのファイルシステムを使用します。使用できる File プラグインは 1 つだけです。

### サーバーの呼び出しに対する応答

File プラグインは非同期です。サーバーがプラグインを呼び出したときに、プラグインは直ちには応答しません。サーバーがプラグインを呼び出すと、プラグインは応答インターフェイスによってサーバーに呼び出しを返します。

サーバーがプラグイン上の関数を呼び出したときに、プラグインインターフェイスはその操作が失敗したことを示すエラーコード (-1) を返すことができます。関数がエラーを返す場合、プラグインがサーバーに呼び出しを返さないようにする必要があります。

プラグインへの呼び出しが正常に戻る (0 が返される) 場合、プラグインはサーバーに呼び出しを返し、サーバーから受け取ったコンテキストを渡す必要があります。

close() および remove() の呼び出しは、この規則に対する例外です。サーバーがプラグイン上の close() メソッドと remove() 関数を呼び出した場合、サーバーは応答に関与しません。コールバックが発生する前に要求に関連付けられているリソースを解放するために、サーバーは pCtx ポインタとして NULL を渡すことがあります。コンテキストが NULL になっている場合、サーバーに呼び出しを返す必要はありません。プラグインが NULL コンテキストとともにサーバーに呼び出しを返した場合、サーバーはそれを無視します。

## 検証のための外部 SWF ファイルの取得

SWF ファイルを検証することで、許可されたアプリケーションだけが Flash Media Server の特定のインスタンスにアクセスできるように制限されます。この方法によって、リソースをストリーミングしようとするアプリケーションが第三者によって独自に作成されないようにします。

Content Distribution Network では、検証する SWF ファイルは外部コンテンツリポジトリ、またはクラスタ内の別のサーバーにあります。サーバーで検証するために、File プラグインを使用して、外部の場所に格納されている SWF ファイルを取得することができます。File プラグインを使用することによって、SWF ファイルを Flash Media Server にローカルに保存する必要がなくなります。開発者は、サーバーに影響を与えずに SWF ファイルを頻繁に更新することができます。File プラグインを使用することで、開発者によるコンテンツ管理がシンプルかつ容易になります。

File プラグインを使用する場合、アプリケーションの種類に関わらず、サーバーは SWF ファイルをファイルレベルで検証します。SWF の検証は、File プラグインを通じてアプリケーション単位で有効にすることはできません。

**注意:** File プラグインを使用して SWF ファイルを取得できるのは、バージョン 1.0 のプラグイン (Flash Media Server 3.5 以降) だけです。

File プラグインによって SWF ファイルを検証するには、次のワークフローに従います。

- Application.xml で SWF 検証の有効化
- Server.xml で SWF 検証の有効化
- File プラグインの実装

### Application.xml で SWF 検証の有効化

1 XML エディタで Application.xml を開きます。

2 SWFVerification タグの enabled 属性を true に設定します。

```
<SWFVerification enabled="true">
```

このタグによって、そのファイルが vhost レベルの XML ファイルでもアプリケーションレベルの XML ファイルであっても、すべてのアプリケーションの検証が可能になります。

3 SWFFolder タグは空のままにします。この設定により、デフォルトの SWFFolder の値がサーバーからプラグインに渡されます。デフォルト値は、アプリケーションのフォルダの下の SWF フォルダです。例えば、applications/application\_name/SWFs です。

File プラグインは、この値を外部のコンテンツリポジトリにリダイレクトする必要があります。

4 (オプション) 追加の検証タグを設定します。

## Server.xml で SWF 検証の有効化

- 1 XML エディタで Server.xml を開きます。
- 2 Plugins タグで、FilePlugin タグの enabled 属性を true に設定します。

```
<Plugins>
  <FilePlugin enabled="true">
    <Content type="Streams">true</Content>
    <Content type="SWF">false</Content>
  </FilePlugin>
</Plugins>
```

- 3 SWF ファイルの検証を有効にするには、<Content type="SWF"> 属性を true に設定します。
- 4 (オプション) グローバル検証を設定するには、SWFFolder タグを空のままにします。それによって、デフォルトの SWFFolder の値がサーバーからプラグインに渡されます。File プラグインは、この値を外部のコンテンツリポジトリにリダイレクトする必要があります。  
グローバル検証では、すべてのアプリケーションに共通の SWF ファイルのグループに対する検証を設定できます。
- 5 (オプション) その他の検証設定を設定します：DirLevelSWFScan、MaxNumberofRequests および UserDefined エメント内の必要なユーザー定義キー。『設定および管理ガイド』の「XML リファレンス」を参照してください。

## File プラグインの実装

File プラグインは外部の場所にあるディレクトリまたはファイルを開き、ディレクトリおよびファイルのリストをサーバーに渡すことができます。File プラグインには、SWF ファイルからストリームファイルを識別する特定の属性が含まれていません。サーバーは、サーバー上に SWF ファイルが格納されている場合と同様に検証を実行します。

サーバーに含まれている File プラグインのサンプルを参照してください。サンプルではコードの主要部分を示しています。

File プラグインのサンプルファイルには、IFmsFileAdaptor::open を呼び出す関数が含まれています。open() 関数には、sFileName パラメータが含まれています。sFileName の値はサーバーにより File プラグインに渡されます。これは Application.xml または Server.xml の SWFFolder タグのデフォルト値です。

File プラグインは、SWFFolder の値を外部コンテンツリポジトリにリダイレクトします。サンプルではそのリポジトリに対するハンドルを作成し、サーバーに渡しています。getAttributes() 関数はさらにパラメータとして sFileName を取ります。

open() 関数を呼び出すと、sFileName の値がファイルであるかディレクトリであるかを判定するコードがサンプルに含まれて、ファイルが開きます。ディレクトリが開き、一時ファイルに書き込まれます。サンプルでは、新しい属性 FMSFileAttribute::kMode を使用して、sFileName がファイルであるかディレクトリであるかが判定されます (ストリームは 1、ディレクトリは 0)。

File プラグインは、この一時ファイルに対するハンドルをサーバーに返します。ディレクトリにサブディレクトリまたは SWF ファイルが含まれている場合、サーバーはディレクトリまたはファイルを開き、すべてのサブディレクトリとファイルが読み取られるまでプロセスを続行します。サーバーはプラグインの read() メソッドを使用してファイルを読み取り、検証を実行します。サーバーはプラグインの close() メソッドを使用して、完全に読み取ったファイルを閉じます。

# 索引

## 記号

.NET 2003 2, 5

## A

Access プラグイン

サーバーへの接続 6

説明 6

フォルダ権限の設定 8

Application.xml ファイル 8, 22

Authorization プラグイン

エッジサーバー、デプロイ 11

クライアント統計へのアクセス 11

コード 9

説明 8

## F

File プラグイン

SWF ファイル 22

実装 23

説明 21

Flash Media Server 2 6

FmsMedia.h ファイル 4

## G

GNU Compiler Collection 2

## H

HTTP ファイルアクセス 21

## I

IFCAccessAdaptor.h ファイル 4

I/O 21

## L

Logger.xml ファイル 5

## S

Server.xml ファイル 5, 23

SWF ファイル 22

## V

Visual C++ 2005 2

## X

XML 設定ファイル、データの取得 4

## い

イベント、サーバーへのアクセス 8, 12

インストール用フォルダ 4

## く

クライアントプロパティ、アクセス 7, 12

## こ

コンテンツ配信ネットワーク (CDN) 22

## さ

サーバーの起動 4

サーバーの停止 4

サンプルファイル 2

## す

ストリームのパス、マッピング 10

ストリームのパスのマッピング 10

## せ

セキュリティ 6

設定ファイル、データの取得 XML 5

## た

タイムクリティカルな呼び出し 5

## ふ

負荷分散 7

プラグイン

アップグレード 1, 5

開発 2

デプロイメント 4

バージョン 1

プラグインのコンパイル 2

プラグインのデプロイメント 4  
プロパティ、クライアントへのアクセス 7, 12

## へ

ヘッダーファイル 4

## ろ

ログイン 4